3NF

We're going along with 3NF because we want a lot of the referential information about storms to be quickly accessible, which the often faster query performance of 3NF should be able to provide. The form is additionally beneficially lax about dependencies, which is essential because our current database schema is largely constructed to meet the criteria of a one-to-one and one-to-many relationship, the latter of which creates a lot of informational redundancy between different tables because specific statistics about weather events (e.g. property damage) are present between wildly different kinds of weather.

3NF Form

In these tables, the primary key of each table ensures the uniqueness of the record, and all attributes in a record are a function of the primary key only, satisfying the requirements of 3NF.

1)
User_id -> user_name, password
event_id -> category_id, category_name
tornado_id-> temperature, wind_speed, ef_scale, tornado_size
blizzard_id-> temperature, wind_speed, snow_depth
hail_id-> temperature, hailstone_size, fall_speed

2)
Remove transitive dependencies

- category_name depends on category_id
- category_id depends on event_id

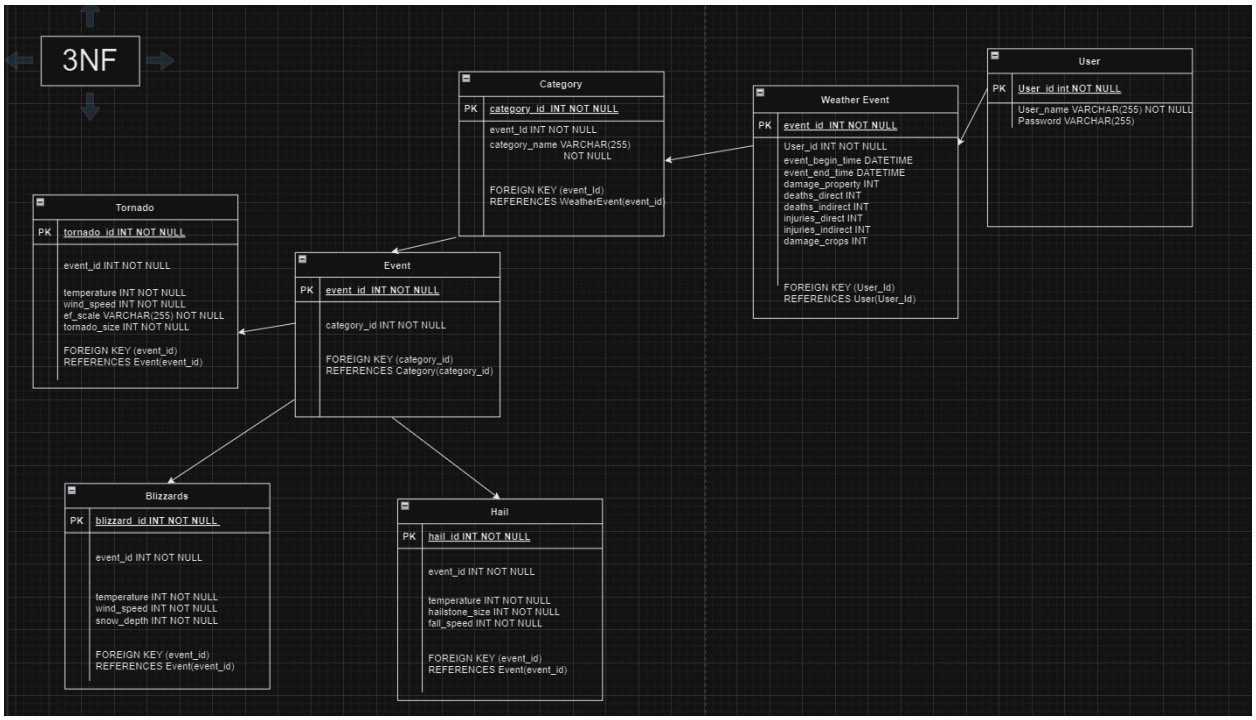
We can split these into 2 different dependencies
event_id -> category_id
category_id -> category_name

Tornado Table, Blizzard Table, Hail Table, is in 3NF: There are no transitive dependencies
3)

Final 3NF Form:

User_id -> user_name, password
event_id -> category_id
category_id -> category_name
tornado_id-> temperature, wind_speed, ef_scale, tornado_size
blizzard_id-> temperature, wind_speed, snow_depth
hail_id-> temperature, hailstone_size, fall_speed

3NF Form Visualized:



Assumptions

To ensure that the database schema is in 3NF, we need to make certain assumptions about the relationships and entities involved. Here are the assumptions based on the entities and relationships you've described:

Entities and Their Assumptions:

1. User
   - Each user is uniquely identified by `User_id`.
   - The `username` and `password` are unique to each `User_id` and do not depend on any other attribute.

2. Event
   - Each event is uniquely identified by `event_id`.

3. Category
   - Each category is uniquely identified by `category_id`.
   - The `category_name` is unique to each `category_id` and does not depend on any other attribute.

4. Tornado
   - Each tornado event is uniquely identified by `tornado_id`.
   - The attributes `temperature`, `wind_speed`, `ef_scale`, and `tornado_size` are dependent solely on `tornado_id`.

5. Blizzard
   - Each blizzard event is uniquely identified by `blizzard_id`.
   - The attributes `temperature`, `wind_speed`, and `snow_depth` are dependent solely on `blizzard_id`.

6. Hail
   - Each hail event is uniquely identified by `hail_id`.
   - The attributes `temperature`, `hailstone_size`, and `fall_speed` are dependent solely on `hail_id`.

Relationships and Their Assumptions:

1. User-Event Relationship
   - If a relationship exists, we assume that each event is related to a single user, making `User_id` a foreign key in the Event table.

2. Event-Category Relationship
   - `category_name` is functionally dependent on `category_id`, which is a different entity from the Event. This is the transitive dependency that needs to be removed for 3NF.

3. Specific Weather Event Tables (Tornado, Blizzard, Hail)
   - These are tables that inherit `event_id` as a foreign key, indicating a one-to-one relationship with specific records in the Event table.
   - Each of these tables has attributes that are uniquely determined by their respective primary keys tornado_id`, `blizzard_id`, `hail_id`.

5) Convert your conceptual database design (ER/UML) to the logical design (relational schema). A relational schema is not a DDL.

User Table:

User(User_id: INT [PK], user_name: VARCHAR(X), password: VARCHAR(X))

Category Table:
Category(category_id: INT [PK], category_name: VARCHAR(X))

WeatherEvent Table:

WeatherEvent(event_id: INT [PK],
User_id: INT [FK to User.User_id],
category_id: INT [FK to Category.category_id],
eventBeginTime: DATETIME,
eventEndTime: DATETIME,
damageProperty: VARCHAR(X),
deathsDirect: INT,
deathsIndirect: INT,
injuriesDirect: INT,
injuriesIndirect: INT,
damageCrops: VARCHAR(X))

Tornado Table:

Tornado(tornado_id: INT [PK],
event_id: INT [FK to WeatherEvent.event_id],
temperature: DECIMAL,
wind_speed: INT,
ef_scale: VARCHAR(X),
tornado_size: DECIMAL)

Blizzard Table:

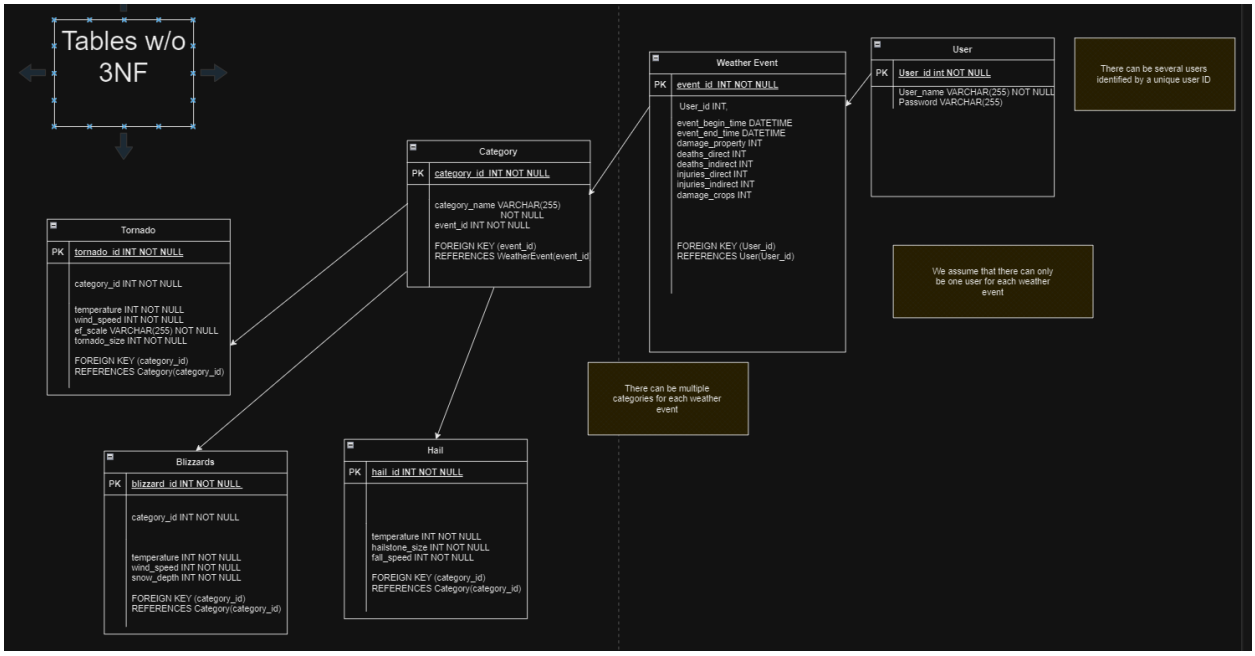
Blizzard(blizzard_id: INT [PK],
event_id: INT [FK to WeatherEvent.event_id],
temperature: DECIMAL,
wind_speed: INT,
snow_depth: DECIMAL)

Hail Table:

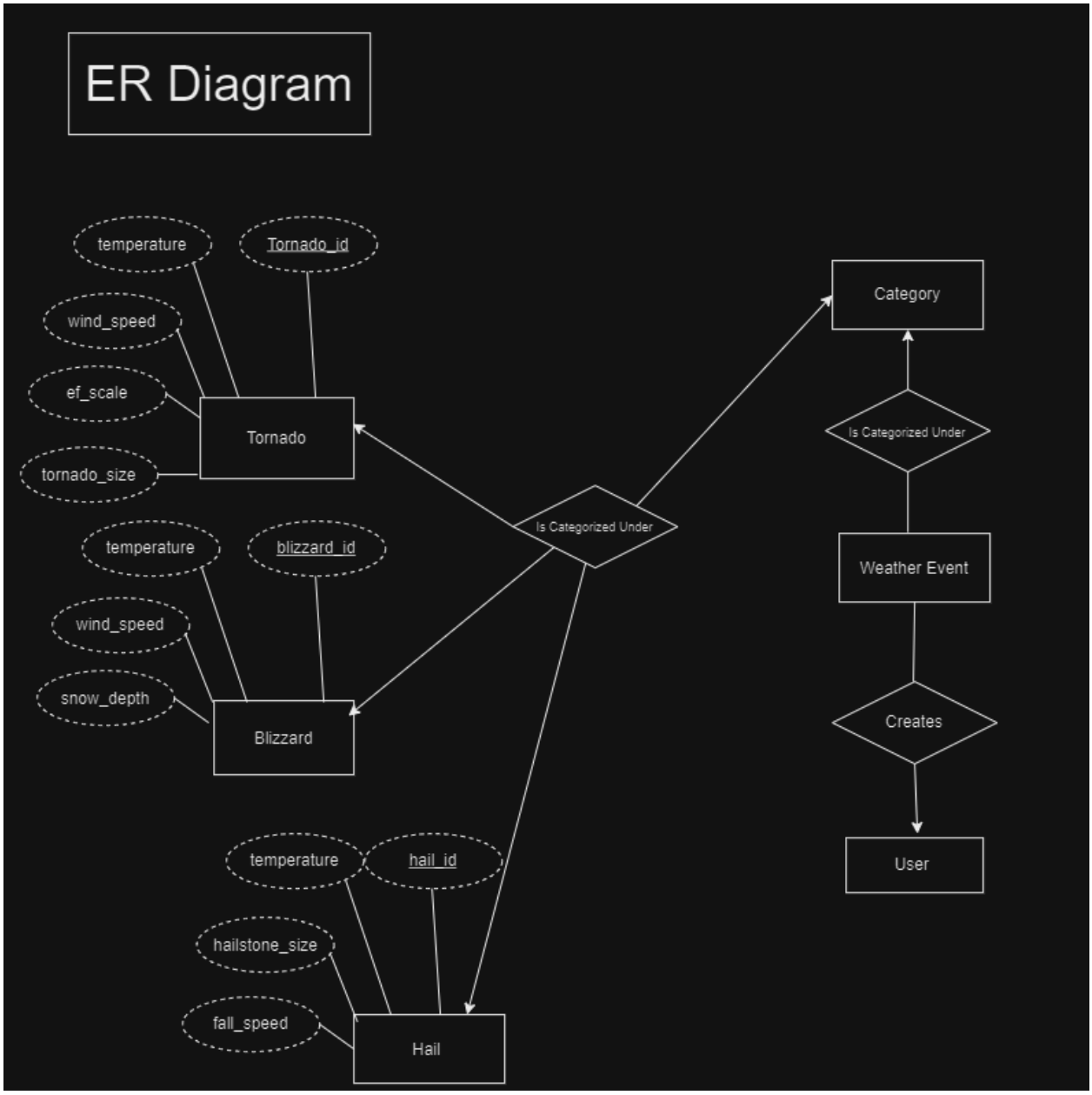
Hail(hail_id: INT [PK],
event_id: INT [FK to WeatherEvent.event_id],
temperature: DECIMAL,
hailstone_size: DECIMAL,
fall_speed: DECIMAL)

Diagrams below

Original:



ER equivalent:


ER Diagram
temperature
Tornado_id
wind_speed
ef_scale
Tornado
tornado_size
temperature
blizzard_id
wind_speed
snow_depth
Blizzard
temperature
hail_id
hailstone_size
fall_speed
Hail
Is Categorized Under
Category
Is Categorized Under
Weather Event
Creates
User